CS M151B Homework 4
Baolinh Nguyen
UID: 104732121

4.7 In this exercise we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word:

10101100011000100000000000010100.

Assume that data memory is all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

r0	r1	r2	r3	r4	r5	r6	r8	r22	r31
0	-1	2	-3	-4	10	6	8	2	16

4.7.1 [5] <§4.4> What are the outputs of the sign-extend and the jump "Shift left 2" unit (near the top of Figure 4.24) for this instruction word?

4.7.2 [10] <§4.4> What are the values of the ALU control unit's inputs for this instruction?

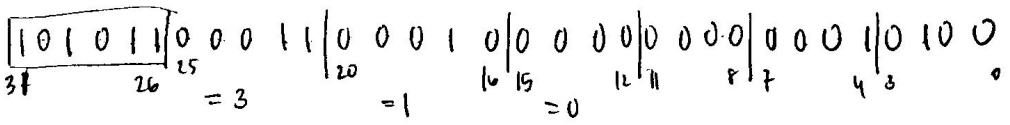
4.7.3 [10] <§4.4> What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.

4.7.4 [10] <§4.4> For each Mux, show the values of its data output during the execution of this instruction and these register values.

4.7.5 [10] <§4.4> For the ALU and the two add units, what are their data input values?

4.7.6 [10] <§4.4> What are the values of all inputs for the "Registers" unit?

10 (011 - sw inst.



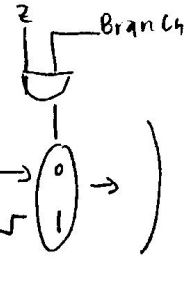
4.7.1 [25-0] : 0 0 0 11 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0

shift left 2

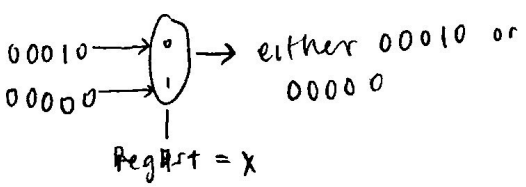
0 0 0 11 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0

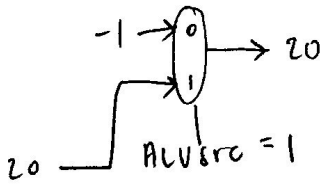
4.7.2. ALU control unit's inputs 0 0 1 0

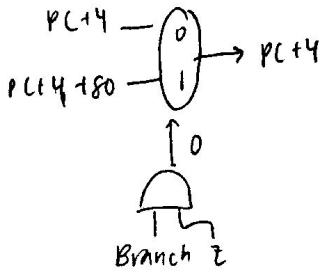
4.7.3. The value of the PC is PC+4. The PC goes to the ALU, gets 4 added to it (PC → ALU ← 4 → ∝) & then enters a MUX (∝ → 0, ∝ → 1 →) but because Branch = 0, the value for 0 is taken, which is PC+4.



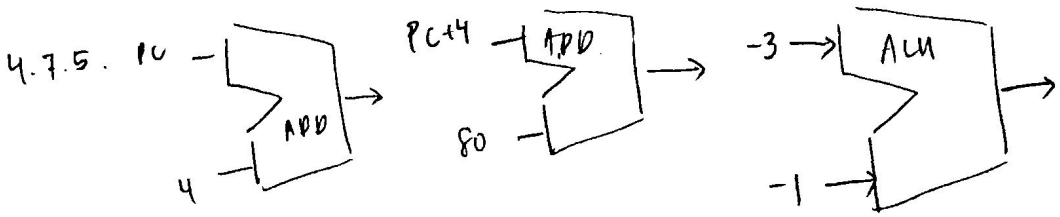
4.7.4.



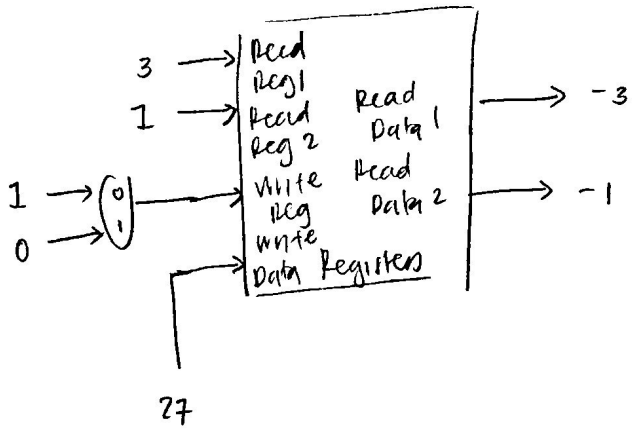




4.7.5.


PC
ADD
4
PC+4
ADD
80
-3
ALU
-1

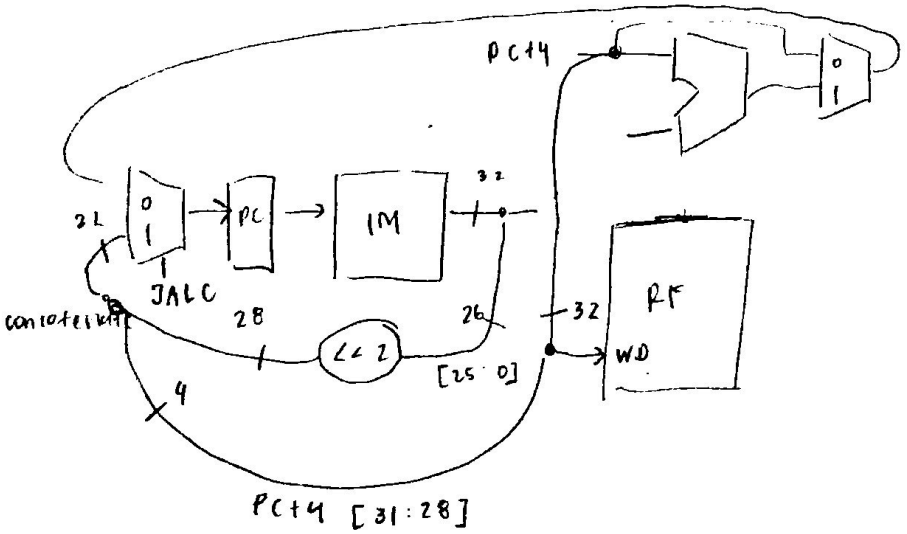
4.7.6.


3
1
1
0
0
1
Read Reg1
Read Reg 2
Write Reg
Write Data
Registers
Read Data 1
Read Data 2
-3
-1
27

CS M151B Homework 4
Baolinh Nguyen
UID: 104732121

2) jal instruction - J type instruction that has the following effect:

R[$r31] = PC+4
PC = [31..28](PC+4) | [27..0] (I<<2)
(in other words, this works exactly like the jump instruction we covered but also writes PC+4 to register $r31 - and always writes to register $r31)

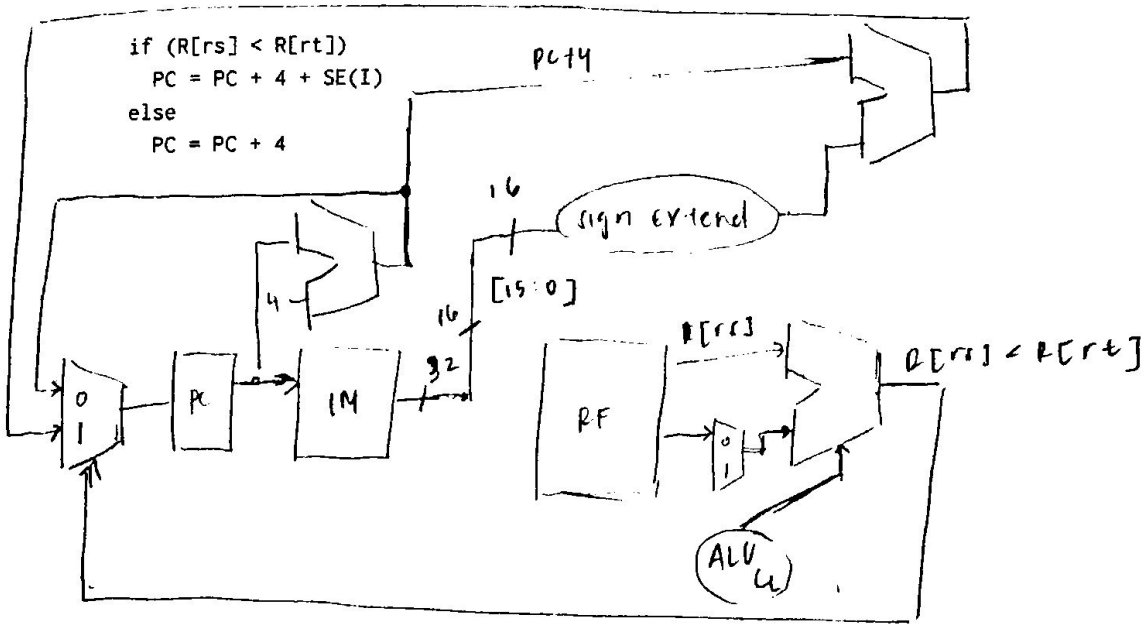


Ct	JAL
REGDST	X
ALUSRC	X
MEM TO REG	X
REGWRITE	1
MEMWRITE	0
MEMREAD	0
BRANCH	X
ALUOP	X
JALC	1

CS M151B Homework 4
Baolinh Nguyen
UID: 104732121

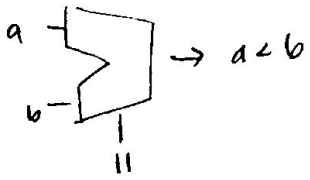
1) blt instruction - I type instruction that has the following effect:

```
if (R[rs] < R[rt])
  PC = PC + 4 + SE(I)
else
  PC = PC + 4
```



ALUOP

ALUOP	operation	ALU
00	X	add
01	X	sub
10	add	add
	sub	sub
	and	and
	or	or
11	set on less than	set on less than



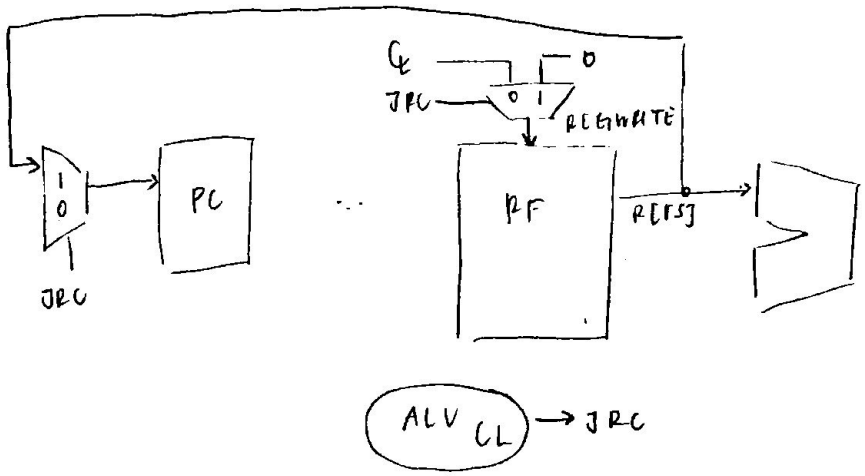
CL

	BLT
REGDST	X
ALUSRC	0
MEMTOREG	X
REGWRITE	0
MEMWRITE	0
MEMREAD	0
BRANCH	0
ALUOP	11

CS M151B Homework 4
Baolinh Nguyen
UID: 104732121

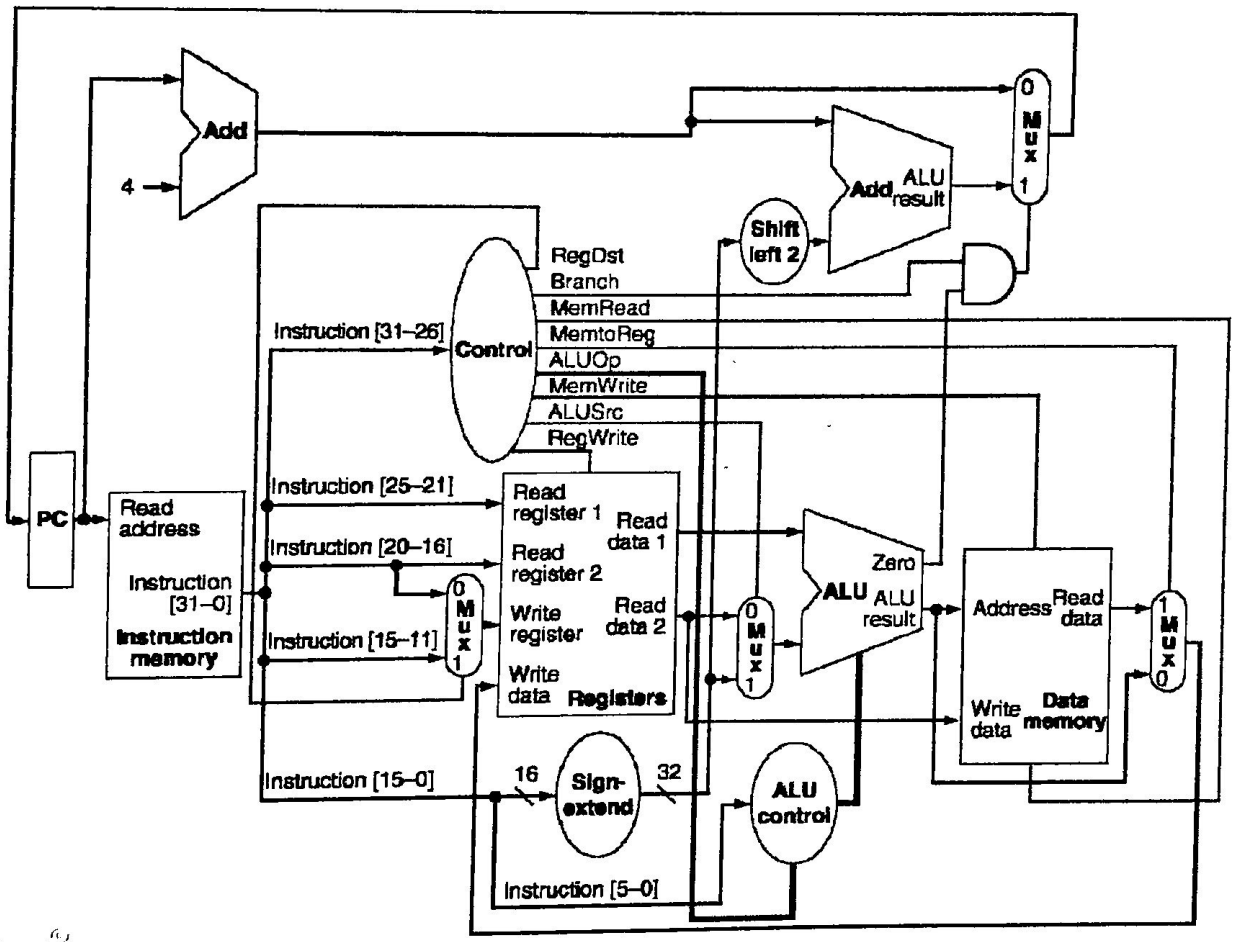
3) jr instruction - R type instruction that has the following effect:

PC = R[rs]



ALU OP	FUNC	ALU	JRC
00	X	add	0
01	X	sub	0
10	add	add	0
	sub	sub	0
	and	and	0
	or	or	0
	jrc	X	1

Datapath With Control



ALU Control

- Assume 2-bit ALUOp derived from opcode
  - Combinational logic derives ALU control

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111